SISTEMAS DISTRIBUIDOS – GRUPO A

ENTREGABLE1 LAB

INDICE

REPOSITORIO GITHUB
https://github.com/jupcan/zeroc-ice

COMPONENTES DEL GRUPO
Juan Perea Campos

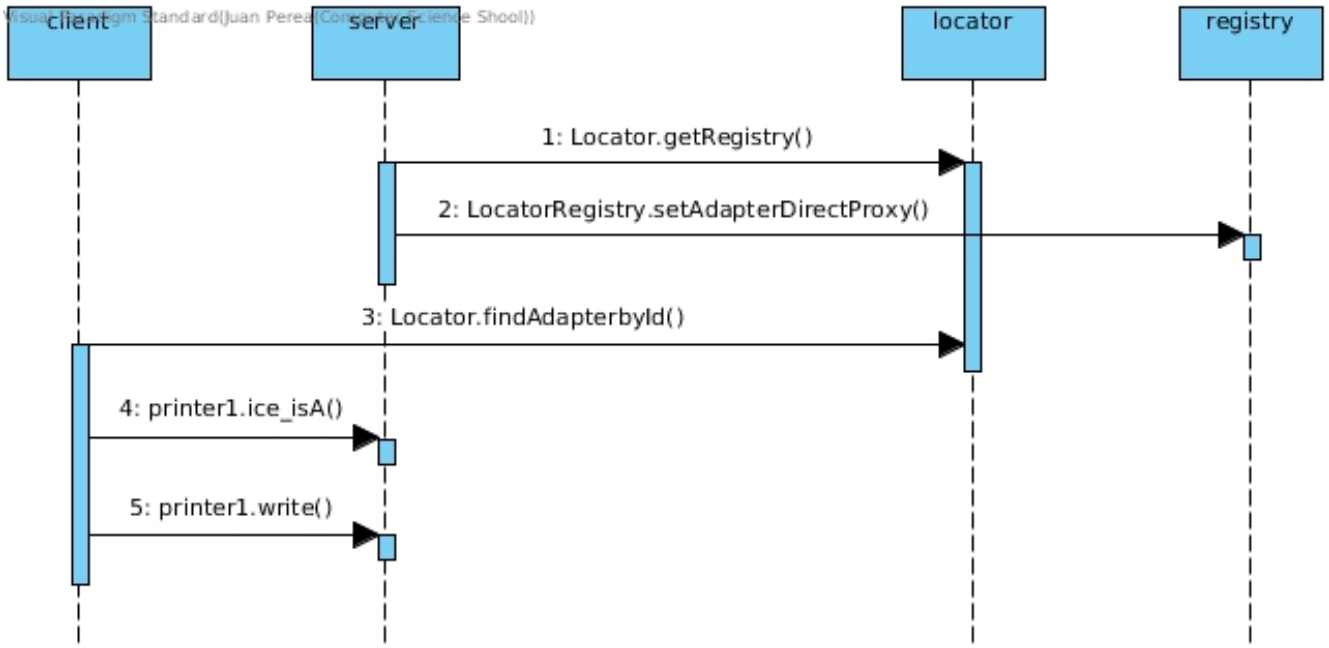
1. DOCUMENTACIÓN ENTREGABLE1

1.1 SECUENCIA DE MENSAJES

De forma redactada, la secuencia de mensajes que tiene lugar, filtrando por el protocolo **icep** la captura adjunta en el entregable realizada con wireshark, es la siguiente:

1. El servidor pregunta al locator por el registry para saber de su disponibilidad.
2. Una vez lo sabe, registra el adaptador en el registry.
3. Posteriormente el cliente pregunta al locator por el id del adaptador que lo conoce al estar ya registrado y puede devolverle el proxy directo hacia el mismo.
4. Se consulta con el servidor si printer1 es un objeto printer.
5. En caso afirmativo del paso anterior, sabiendo que está en uso un proxy indirecto y una vez obtenido su correspondiente proxy directo para establecer la comunicación, se llama al write para imprimir el resultado en el servidor.

He decidido a su vez realizar un diagrama de secuencia para una mejor visualización del flujo de mensajes, así como de la fuente y el contenido de cada uno de ellos.



Utilizando el manual disponible en la web de ZeroC podemos explicar más detalladamente cada una de las llamadas realizadas por las fuentes de los mensajes.

1. **Locator.getRegistry()**: método que pertenece a la interfaz del locator de Ice. "*Usada por clientes para buscar adaptadores y objetos. También utilizada por servidores para obtener el proxy del registry asociado al locator."* En nuestro caso, nos quedaríamos con lo último pues es el servidor quién lo utiliza. Sabemos que el destino es el locator por el nombre intuitivo además del puerto de destino 9999.

2. **LocatorRegistry.setAdapterDirectProxy(id, proxy)**: método perteneciente a la interfaz del registry perteneciente al locator de Ice y que es usada por servidores para registrar *endpoints* de adaptadores a través del locator. Tiene como parámetros el id del adaptador (*PrinterAdapter1*), y el proxy (*tcp*) que contiene todos los *endpoint* asociados a ese id previo y que podrán ser consultados por el locator.

3. **Locator.findAdapterbyId(id)**: al igual que el *.getRegistry()*, pertenece a la interfaz del locator de Ice y, en este caso, nos quedamos con la primera parte de la cita textual ya que es el cliente que lo invoca para buscar el adaptador dado un id (*PrinterAdapter1*). Devuelve *null* si no existe o el proxy directo del adaptador en caso contrario que es lo que sucedería en nuestro ejemplo.

4. **printer1.ice_isA()**: método para comprobar si *printer1* es un adaptador de objetos del tipo *printer* previamente definido, debe ser del tipo boolean.

5. **printer1.write("hola mundo")**: si el paso 4 es *true*, se puede construir el mensaje de invocación y éste puede ser enviado de la forma habitual al servidor.